JACKSON LAWRENCE
00000070612

LOUIS GABRIEL HERNANDES
00000070250

# SIMULATED ANNEALING

OSCAR JIRO HARLISON
00000072786

IGNATIUS STEVEN
00000070642

# DEFINITION

- **Simulated Annealing (SA)** adalah sebuah algoritma metaheuristik untuk memecahkan masalah-masalah optimasi, tepatnya menemukan global optimum dari sebuah fungsi
- Dikenalkan oleh Kirkpatrick et al. pada 1983 untuk menjawab Traveling Salesman Problem, SA terinspirasi dari perilaku atom pada **proses annealing atau pemanasan logam** pada bidang metalurgi yang dilakukan untuk mengubah properti fisika logam
- SA juga merupakan adaptasi dari Metropolis–Hastings oleh Metropolis et al. pada 1953, sebuah metode Monte Carlo untuk mendapatkan sampling

# ANNEALING PROCESS

- Dalam proses annealing di metalurgi, sebuah logam **dipanaskan ke temperatur yang tinggi secara cepat**, lalu **didinginkan secara berkala**
- Pada temperatur yang tinggi, atom-atom logam juga bergerak dengan cepat dan saat temperatur sudah diturunkan, energi kinetik pada atom-atom pun juga berkurang
- Pada akhir proses annealing, atom-atom pada logam berada di sebuah crystalline structure atau **kondisi yang lebih teratur**, sehingga material logam lebih **lentur dan mudah dipakai atau dikerjakan**

# WHY USE SA?

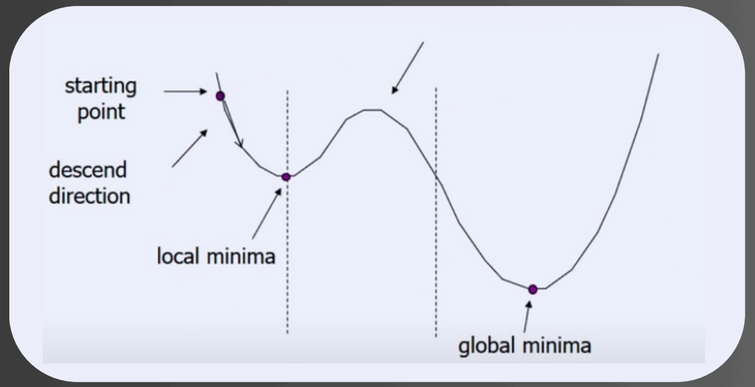
- **Simpel dan fleksibel** karena dapat diaplikasikan di banyak jenis masalah optimasi
- Dapat **lolos dari local optimum** dan menemukan global optimum secara baik karena mampu menerima solusi yang lebih buruk
- **Tidak memerlukan kalkulasi matematika yang kompleks** seperti informasi gradien pada fungsi objektif seperti gradient descent

# APPLICATIONS

- VLSI Design
- Multiprocessor scheduling
- Crystal structure prediction
- Traveling salesman
- Vehicle routing
- Scheduling problems

# HOWEVER...

- **Cooling schedule perlu diatur secara baik:** apabila terlalu cepat, SA berisiko konvergensi menuju local optimum; apabila terlalu lambat, SA dapat membutuhkan waktu konvergensi yang terlalu lama
- **Memerlukan banyak iterasi untuk solusi yang baik**, terutama apabila masalahnya kompleks
- **Tidak memanfaatkan heuristik atau bimbingan yang problem-specific** sehingga dapat lebih inefisien dibandingkan algoritma lain yang lebih dispesialisasi




*Harus kabur dari terjebaknya di local minima

# HOW IT WORKS

- Awalnya, algoritma dimulai dengan **temperatur tinggi** dan suatu konfigurasi molekul logam yang merepresentasikan **solusi awal random**
- Di setiap iterasi, **solusi baru akan dibuat** dengan membuat perubahan dari sebelumnya, dan **temperatur sistem akan perlahan diturunkan**
- Saat temperatur masih tinggi, solusi lebih buruk memiliki probabilitas lebih tinggi untuk diterima
- Saat temperatur sudah merendah, sistem akan lebih bercondong ke solusi yang lebih baik, layaknya hill-climbing klasikal
- Dalam kata lain, algoritma ini **rela menerima kondisi yang buruk di awal** demi mendapat yang **terbaik di akhir**
- Dengan demikian, semestinya suatu crystalline structure atau **solusi optimal akan didapat di akhir** algoritma
- Solusi global optima dapat diraih apabila cooling proses semakin melambat

# OVERALL CONCEPT


| Physical Annealing | Simulated Annealing |
|---|---|
| Metal | Optimization Problem |
| Energy State | Cost Function |
| Temperature | Control Parameter |
| Crystalline Structure | The Optimal Solution |

# KEY COMPONENTS & STOPPING CRITERIA

## Temperature

01

Sebuah parameter yang membimbing eksplorasi search space; lebih tinggi mempromosikan eksplorasi, rendah mempromosikan eksploitasi

## Cooling Schedule

02

Mengatur bagaimana temperatur diturunkan, umumnya dapat secara geometris atau linear

## Perturbation Function

03

Sebuah fungsi yang membuat solusi baru dari neighborhood (kumpulan solusi yang dapat dicapai dengan mengubah sedikit dari solusi sebelumnya)

## Metropolis Criterion

04

Peraturan yang menentukan kemungkinan solusi baru diterima walaupun solusi baru lebih buruk dari sebelumnya berdasarkan temperatur dan kualitas solusi; solusi baru yang lebih baik akan selalu diterima

## Boltzmann Distribution

05

Sebuah metode distribusi yang hanya terjadi untuk menangkap ide dari menerima hasil baru yang buruk dan mereka tersedia bukan karena solusi buruk yang ekstrem dibanding dengan solusi sekarang

## Stopping Criteria

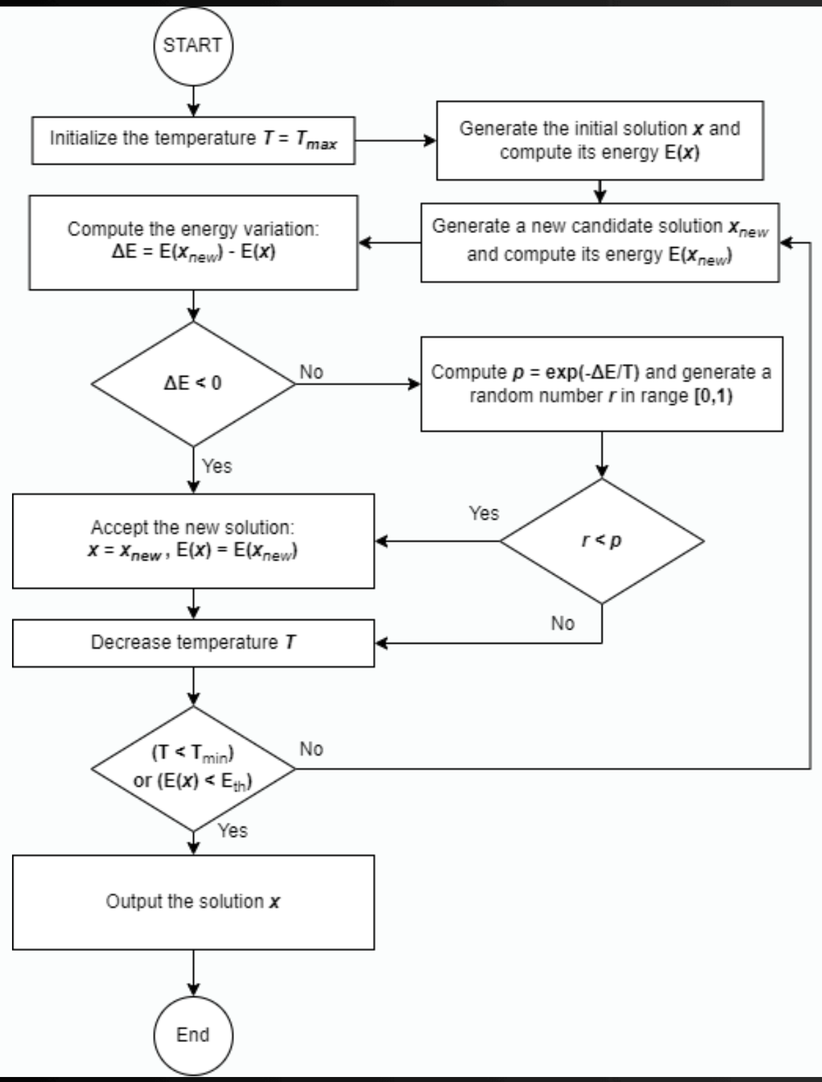
Rata-rata perubahan nilai fitness lebih kecil dari toleransi (dianggap telah berkonvergensi)

Temperatur sudah mendekati nol.

Sudah melebihi batas iterasi maksimum.

Sudah melebihi batas waktu.

# PSEUDOCODE




**Algorithm 1:** Simulated Annealing Algorithm

**Input:** $T_0$, $c$, $itmax$
**Output:** $x_{\text{opt}}$, $f_{\text{best}}$

Bangkitkan $x_0$;
$f_{\text{best}} = f(x_0)$;
$T = T_0$;
$it = 1$;
**while** *kritera penghentian belum dicapai* **do**
  **if** $it < itmax$ **then**
    Bangkitkan $\Delta x$ berdistribusi Gaussian;
    **if** $f(x_0 + \Delta x) < f(x_0)$ **then**
      $f_{\text{best}} = f(x_0 + \Delta x)$;
      $x_0 = x_0 + \Delta x$;
    **else**
      $\Delta f = f(x_0 + \Delta x) - f(x_0)$;
      Bangkitkan bilangan random $r$;
      **if** $r < \exp\left(\frac{-\Delta f}{kT}\right)$ **then**
        $x_0 = x_0 + \Delta x$;
  $it = it + 1$;
$T = T * c$;

https://github.com/oscarjiro/simulated-annealing-example

# PARAMETER TUNING

| Parameter | Exploration | Exploitation |
|---|---|---|
| Initial Temperature ($T_0$) | Higher Temperature leads to more exploration | Lower Temperature accelerates exploitation |
| Cooling Schedule ($c$) | Slow Cooling promotes extensive exploration | Fast Cooling focuses on early exploitation |
| Number of Iteration ($it_{max}$) | Longer run times provide more opportunities for exploration | Premature stopping focuses on immediate exploitation |

**Complex problem may require more exploration but simple problem will benefit more from exploitation**

# COOLING SCHEDULE

| Schedule | Formula | Cooling Rate | Characteristics |
|---|---|---|---|
| Geometric Cooling | $T_{k+1} = \alpha \cdot T_k$ | Moderate | Most common, balance between speed and exploration |
| Linear Cooling | $T_{k+1} = T_k - \beta$ | Moderate to Fast | Constant decrease, simple but needs good parameter tuning |
| Exponential Cooling | $T_{k+1} = T_0 \cdot e^{-\lambda k}$ | Fast | Faster cooling at the start, slows down later |
| Logarithmic Cooling | $T_{k+1} = \frac{T_0}{\log(k+1)}$ | Slow | Slow cooling, allows extensive exploration |
| Quadratic Cooling | $T_{k+1} = \frac{T_0}{(k+1)^2}$ | Very Fast | Rapid cooling, can lead to early convergence |
| Adaptive Cooling | Varies based on improvement | Varies (based on progress) | Flexible, adjusts cooling based on optimization progress |

# TIME COMPLEXITY → $O(T * I)$

# SPACE COMPLEXITY → $O(M * N * K)$

**Informations:**

- I : Number of iterations
- T : Temperature
- N : Number of array
- M : Dimension
- K : Additional Space

# DISTRIBUSI PROBABILITAS BOLTZMANN

$$P(E) = e^{\frac{-E}{k \times T}}$$

# KRITERIA METROPOLIS

$$\Delta E = E_{i+1} - E_i$$ = A

$$r \leqslant e^{\frac{-\Delta E}{k \times T}}$$ = B

$$|x_{i+1} - x_i| < \varepsilon$$ = C

**Informations :**

- k = Boltzmann constant = 1.380649x10$^{-23}$J/K
- e = Euler's number = 2.71828
- $x_i$ = $i^{th}$ vertex
- ΔE = Energy / Value differences
- r = Random value = randn(0,1)
- T = Temperature
- ε = Termination criterion
- σ = Standard deviation
- $E_i$ = This solution & $E_{i+1}$ = Neighborhood solution

# ADDENDUM TO DETERMINE NEXT POINT

$$x = \underset{i+1}{x}_{i} + \sigma \left[ \sum_{i=1}^{n} R_i - \frac{N}{2} \right]$$

$$\sigma = \frac{max - min}{6}$$

**Informations :**

- $R_i$ = Random value
- N = Number of all element or $R_i$

# COOLING SCHEDULE → STEPS

$$T_{new} = T_{old} \times c$$

c = Temperature factor

**STEP-1:** Choose initial point ($x_0$), termination criterion (ε). Set initial temperature to high value (T), and number of iterations (n)

**STEP-2:** Calculate neighboring point randomly

**STEP-3:** If A < 0, set t = t + 1
else create a random number r = rand(0,1)
If B true, set t = t + 1
else go to **STEP-2**

**STEP-4:** If C true and T is small. Return E
else if (t mod n) = 0, then lower T. Go to **STEP-2**

# OPTIMIZING A FUNCTION

Himmelblau function: $f\left(\overrightarrow{x}\right) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$ for $x_1, x_2 \in (0, 5)$

## I. Initialize parameter & Choose search domain (0, 5) for $x_1$ and $x_2$

**Initial Point**

Take mid-point : $\overrightarrow{x_0} = (2.5, 2.5)$

**Termination factor = Permissible error**

$\varepsilon = 0.001$

**Temperature factor**

c = 0.5

**Max Iteration**

t = 2

**Initial Temp (T) *0th Iteration**

Search domain = (0,0) ; (5,0) ; (0,5) ; (5,5)

$f(x_1,x_2)=(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2$

$f(0,0)=(0+0-11)^2+(0+0-7)^2 = 170$

$f(5,0)=(5^2+0-11)^2+(5+0^2-7)^2 = 200$

$f(0,5)=(0^2+0-11)^2+(0+5^2-7)^2 = 360$

$f(5,5)=(5^2+5-11)^2+(5+5^2-7)^2 = 890$

T = (170 + 200 + 360 + 890)/4 = 405

## II. Generate value point for $x_1$ dan $x_2$, random value, determine next point, & ΔE value

**Value of Initial Point**

$f(x_0) = f(2.5;2.5) = (2.5^2+2.5-11)^2+(2.5+2.5^2-7)^2 = 8.125$

**Generate random value**

Assume $\Delta_1$ = 0.037 & $\Delta_2$ = -0.086

**Determine next point**

$x_1$ = (2.5; 2,5) + (0.037; -0.086) = (2.537; 2.414)

$f(x_1) = (2.537^2+2.414-11)^2+(2.537+2.414^2-7)^2 = 6.482$

**ΔE value**

ΔE = $f(x_1)$ - $f(x_0)$ = 6.482 - 8.125 = -1.643

Because ΔE < 0, then accept new value and increment t=t+1=0+1=1

$||x_1 - x_0|| = \sqrt{(0.037)^2+(-0.086)^2} = 0.101$

Because 0.101 > 0.001, then don't stop & because t mod n = 1 mod 1 = 0, then T = T/2 = 405/2 = 202.5

## III. Repeat Steps 2 for next iteration (0 → 1)

**Generate random value**

Assume $\Delta_1$ = -0.465 & $\Delta_2$ = -1.810

**Determine next point**

$x_2$ = (2.537; 2,414) + (-0.465; -1.810) = (2.072; 0.604)

$f(x_1) = (2.072^2+0.604-11)^2+(2.072+0.604^2-7)^2 = 58.067$

**ΔE value**

ΔE = $f(x_1)$ - $f(x_0)$ = 58.067 - 6.482 = 51.585

Because ΔE > 0, then don't reject straight way. Instead, use Metropolis criterion to decide whether to accept or reject this worse point

$P(E_2) = e^{\frac{-\Delta E}{T}} = e^{\frac{-51.585}{202.5}} = 0.775$

**Generate uniformly distributed number r**

r = 0.649 then r < 0.775 = 0.649 < 0.775 means accept the point & increment t=t+1=1+1=2 (Stopping criteria). **6.482 to 58.067 (Worse Acception)**

# OPTIMIZING A FUNCTION

$$Z = 2 \times x_1 + x_2^2 \quad \text{for} \quad 0 \leqslant x_1, x_2 \leqslant 9$$

## I. Initialize parameter & Choose search domain (0, 9) for $x_1$ and $x_2$

**Initial Temp (T) *Iterasi ke-0**

Search domain = (4,6);(6,4);(4,8);(0,6);(5,7)

$Z = f(x_1,x_2) = 2^* x_1 + x_2^2$

$f(4,6)=2*4+6^2 = 44$

$f(6,4)=2*6+4^2 = 28$

$f(4,8)=2*4+8^2 = 72$

$f(0,6)=2*0+6^2 = 36$

$f(5,7)=2*5+7^2 = 59$

T = (44 + 28 + 72 + 36 + 59)/5 = 47.9

**Multiple T with suitable number to optimize the temperature for more maximize**

T * 10 = 47.9 * 10 = 479 ≈ 500

| Temperature factor | Max Iteration |
|---|---|
| c = 0.5 | t = 2 |

## II. Generate value of initial point for $x_1$ and $x_2$, update T, & Generate next point

**Value of Initial Point = (5, 7) & Update T *Iterasi 0**

$f(x_0) = f(5,7) = 2*5+7^2 = 59$

T_new = T_old * c = 500 * 0.5 = 250

**Generate next point using formula**

$x\_old_{(1)}$ = 5 and σ = (max-min)/6 = (9-0)/6 = 1.5

Generate N and $R_i$ :

- Assume N=4, $R_1$=0.72, $R_2$=0.72, $R_3$=0.15, $R_4$=0.32
- $\sum_{i=1}^{4} R_i = 0.72 + 0.72 + 0.15 + 0.32 = 4.95$

∴ $x\_new_{(1)} = x\_old_{(1)} + \sigma^* (\sum R_i - N/2) = 5 + 1.5^* (4.95 - 4/2) = 4.95$

$x\_old_{(2)}$ = 5 and σ = (max-min)/6 = (9-0)/6 = 1.5

Generate N and $R_i$ :

- Assume N=4, $R_1$=0.32, $R_2$=0.5, $R_3$=0.8, $R_4$=0.4
- $\sum_{i=1}^{4} R_i = 0.32 + 0.5 + 0.8 + 0.4 = 2.02$

∴ $x\_new_{(2)} = x\_old_{(2)} + \sigma^* (\sum R_i - N/2) = 7 + 1.5^* (2.02 - 4/2) = 7.03$

**∴ New point = ($x\_old_{(1)}$, $x\_new_{(2)}$) = (4.05;7.03)**

**$f(4.05;7.03) = 2*4.05+7.03^2 = 59.32$**

## III. Check ΔE, Repeat Steps 2 for next iteration (0 → 1), Update T

**ΔE value for iteration 0**

ΔE = 59.32-59 = 0.32>0 → Go for metropolis algorithm

**Metropolis Algorithm**

$P(E_2) = e^{\frac{-\Delta E}{T}} = e^{\frac{-0.32}{250}} = 0.99$

Assume r = 0.495, then

r < 0.99 = 0.495 < 0.99

**Generate next point using formula**

Using $x\_old_{(1)}$ = 4.05 & $x\_old_{(2)}$ = 7.03

For $x\_new_{(1)}$, assume N=4, $R_1$=0.39, $R_2$=0.22, $R_3$=0.99, $R_4$=0.06

For $x\_new_{(2)}$, assume N=4, $R_1$=0.5, $R_2$=0.2, $R_3$=0.2, $R_4$=0.1

For the same method like in Step 2, new point for iteration 1 is (3.54; 5.53)

∴ $f(3.54;5.53) = 2*3.54+5.53^2 = 39.46$

**Update T**

T_new = T_old * c = 250 * 0.5 = 125

**ΔE value for iteration 1**

ΔE = 39.46-59.32 = -19.82<0 → Accept value and return

**∴ 59 to 59.32 to 39.46 (Most optimal)**

# TRAVELING SALESMAN PROBLEM

| | Kota 1 | Kota 2 | Kota 3 | Kota 4 | Kota 5 |
|---|---|---|---|---|---|
| **Kota 1** | 0 | 132 | 217 | 164 | 58 |
| **Kota 2** | 132 | 0 | 290 | 201 | 79 |
| **Kota 3** | 217 | 290 | 0 | 113 | 303 |
| **Kota 4** | 164 | 201 | 113 | 0 | 196 |
| **Kota 5** | 58 | 79 | 303 | 196 | 0 |

**T_old = 1000**

**T_akhir = 15**

**Ambil rute pertama misalkan rute**

1-2-3-5-4-1 = 132+290+303+196+164 = 1085 = $f_1$

**Konstruksi rute baru dengan menukar 2-3 menjadi 3-2**

1-3-2-5-4-1 = 217+290+79+196+164 = 946 = $f_2$

**∴ Karena $f_2 < f_1$, maka rute 2 diterima**

**Konstruksi rute baru dengan slide 5-4 ke kiri dua kali**

1-5-4-3-2-1 = 58+196+113+290+132 = 789 = $f_3$

**∴ Karena $f_2 > f_3$, maka gunakan kriteria metropolis**

**Cek nilai P(E) dengan r = 0.2**

$P(E) = e^{\frac{-E}{T}} = e^{\frac{-789}{1000}} = 0.4543$

r < 0.4543 = 0.2 < 0.4543 (Accept solusi)

T_new = T_old*0.01= 1000*0.01 = 10 < T_akhir = 10 < 15 (Proses berhenti)

**∴ Solusi terbaik saat ini = 1-5-4-3-2-1**

*f = E

# STEPS IN TSP USING SA

**STEP-1:** Tentukan rute awal secara acak

**STEP-2:** Bangkitkan rute baru dengan 3 metode (Beserta contohnya)

Misalkan ada rute dengan 7 kota yaitu 4-6-2-3-5-1-7

1. Metode membalik (Flip)
   Rute baru menjadi 4-6-1-5-3-2-7
2. Metode menukar (Swap)
   Rute baru menjadi 4-6-1-3-5-2-7
3. Metode menggeser (Slide)
   Rute baru menjadi 4-6-3-5-1-2-7

Cara menentukan antara ketiga metode ini dapat menggunaan **bilangan random r**

- r ≤ 0.33 → Pilih metode swap
- 0.33 ≤ r ≤ 0.67 → Pilih metode slide
- r ≥ 0.67 → Pilih metode flip

**STEP-3:** Jika jarak total rute baru < rute awal, maka rute baru digunakan sebagai urutan terbaik

Jika jarak total rute baru > rute awal, maka gunakan kriteria metropolis untuk menolak / menerima rute baru

**STEP-4:** Jika iterasi maksimum tercapai, maka harus ada penurunan temperatur. Jika suhu sudah lebih kecil dari suhu awal, maka proses berhenti dan dapatkan rute terbaik di proses tersebut. Seiring temperatur turun, maka jumlah pasangan kota untuk ditukar berkurang / Pencarian semakin sempit dan didapatkan rute terbaik dengan jarak total minimum

# TAMBAHAN

**Pada penentuan next point terdapat beberapa cara dari patokan previous point**

## I. Dengan Penambahan bilangan random Δ

Sesuai dengan contoh sebelumnya pada bagian Himmelblau function yaitu :

- Titik awal = $(x_1, x_2)$ = (2.5; 2,5)
- Misalkan $\Delta_1$ = 0.037 & $\Delta_2$ = -0.086
- Titik baru = (2.5; 2,5) + (0.037; -0.086) = (2.537; 2.414)

## II. Dengan Akurasi ±6

Misalkan titik awal = $(x_1, x_2)$ = (4, 5) dan random number = $(u_1, u_2)$ = (0.31; 0.57)

Akurasi ±6 berarti titik awal berada di rentang :
$x_1$ = (4-6, 4+6) = (-2, 10) dan $x_2$ = (5-6, 5+6) = (-1, 11)

Titik baru $x_1$ = -2 + $u_1$*(10-(-2)) = -2+0.31*12 = 1.72
Titik baru $x_2$ = -1 + $u_2$*(11-(-1)) = -1+0.57*12 = 5.84

∴ Titik baru = (1.72; 5.84)

## III. Memakai rumus dan ada hubungannya dengan standard deviation seperti contoh pada fungsi Z sebelumnya

**Links:**

- https://webpages.iust.ac.ir/yaghini/Courses/AOR_891/05_Simulated%20Annealing_01.pdf
- https://www.mathworks.com/help/gads/how-simulated-annealing-works.html
- https://www.eecs.uottawa.ca/ordal/papers/sander/node7.html
- https://saturncloud.io/glossary/simulated-annealing/
- https://sites.gatech.edu/omscs7641/2024/02/19/simulated-annealing-methods-and-real-world-applications/
- https://algorithmafternoon.com/books/simulated_annealing/chapter02/
- https://chatgpt.com/share/66fd2850-7a30-8008-8a90-15d72da2b2d8
- https://cs.stackexchange.com/questions/110238/why-is-the-usual-simulated-annealing-algorithm-using-a-boltzmann-probability

**Paper:**

- https://www.researchgate.net/publication/225260290_The_Theory_and_Practice_of_Simulated_Annealing/link/0c960516543772adef000000/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19

**Video:**

- https://www.youtube.com/watch?v=Yo1F7Z6ha6s&t=471s
- https://www.youtube.com/watch?v=kra3EsSFt2A
- https://www.youtube.com/watch?v=A39YiEfXwec&t=379s
- https://www.youtube.com/watch?v=TC9WNwM2noM
- https://www.youtube.com/watch?v=D2zyF8zhSQ8&t=898s
- https://youtu.be/D2zyF8zhSQ8?si=5Dijjer4hRTLDmpl